Apple *II*

Applesoft BASIC Quick Reference Card

Editing and Cursor Control

LEFT-ARROW	Erase previous character
RIGHT-ARROW	Recopy character under cursor
CONTROL-X	Cancel input line
ESC A	Move right; leave escape mode
ESC B	Move left; leave escape mode
ESC C	Move down; leave escape mode
ESC D	Move up; leave escape mode
ESC I	Move up; remain in escape mode
ESC J	Move left; remain in escape mode
ESC K	Move right; remain in escape mode
ESC M	Move down; remain in escape mode

After ESC, arrow keys are the same as I, J, K, M

ESC E	Clear to end of line
ESC F	Clear to end of screen
ESC @	Clear entire screen; move cursor to top
DEL *n1* , *n2*	Delete from line *n1* to line *n2*

Statements and Lines

Lines typed without a line number are executed immediately; those with a line number are saved for later (deferred) execution.

:	Separates multiple statements on same line
REM	Remarks for human reader

Operations on Whole Programs

NEW	Erase current program, reset all variables
CLEAR	Reset all variables
LIST	Display current program
LIST *n1*-*n2*	Display from line *n1* to line *n2*
RUN	Execute program from beginning
RUN *n*	Execute program starting at line *n*
RUN *name*	Load and execute program name from disk
LOAD	Load program from tape
LOAD *name*	Load program name from disk
SAVE	Save current program on tape
SAVE *name*	Save current program on disk as *name*

Interrupting and Resuming

CONTROL-S	Suspend output (any key to resume)
CONTROL-C	Interrupt program execution
CONT	Continue execution after CONTROL-C, STOP, or END
CONTROL-RESET	Cancel program execution

Variables

Type	Name	Range
Real	AB	+/- 9.99999999 E+37
Integer	AB%	+/- 32767
String	AB$	0 to 255 characters

where A is a letter, B is a letter or digit. Name may be more than two characters, but only first two are significant.

Control

GOTO *n*	Branch to line *n*
ON *expr* GOTO *n1*, *n2*, *n3*, . . .	Branch to line *n1, n2, n3,* . . . depending on value of *expr*
IF *cond* THEN *s1* : *s2* : *s3* : . . .	Execute statements *s1, s2, s3,* . . . if condition cond is true
FOR *v* = *x* TO *y* STEP *z*	Begin loop for all values of *v* from *x* to *y* by *z*; if STEP omitted, 1 is understood
NEXT *v*	Repeat loop for next value of *v*
GOSUB *n*	Branch to subroutine at line *n*
RETURN	Return from subroutine to point of call
ON *expr* GOSUB *n1*, *n2*, *n3*, . . .	Branch to subroutine at line *n1, n2, n3,* . . . depending on value of *expr*
POP	Remove last return address from subroutine stack without branching
ONERR GOTO *n*	Establish error-handling routine beginning at line n
RESUME	Reexecute statement causing error
STOP	Halt execution with message identifying line
END	Halt execution with no message

String Operations

+	Concatenate strings
LEN (*s*)	Length of string *s*
LEFT$ (*s*, *x*)	Leftmost *x* characters of string *s*
MID$ (*s*, *x*, *y*)	*y* characters beginning at position *x* in string *s*
RIGHT$ (*s*, *x*)	Rightmost *x* characters of string *s*
STR$ (*x*)	String representing numeric value *x*
VAL (*s*)	Numeric value of string *s*
CHR$ (*x*)	Character with ASCII code *x*
ASC (*s*)	ASCII code for first character of string *s*

Input/Output

IN# *n*	Accept input from slot *n*
IN# 0	Accept input from keyboard
INPUT *s*; *x*, *y*, *z*	Prompt with string *s*, then read values into variables *x, y, z*; if *s* omitted, ? is used
GET *c*	Read one character into variable *c*
READ *x*, *y*, *z*	Read values from DATA list into variables *x, y, z*
DATA *x*, *y*, *z*	Add values *x, y, z* to DATA list
RESTORE	Restart DATA list from beginning
RECALL *a*	Read array *a* from tape
PDL (*n*)	Read dial of hand control *n*
PR# *n*	Send output to slot *n*
PR# 0	Send output to display screen
PRINT *x*, *y*, *z*	Display or print values *x, y, z*
STORE *a*	Write array *a* to tape
TEXT	Display text
HOME	Clear screen and send cursor to top
;	Start next item at cursor position
,	Start next item at next tab position
SPC (*x*)	Display or print *x* spaces (PRINT statement only)
TAB (*x*)	Move cursor to column *x* (PRINT statement only)
HTAB *x*	Move cursor to column *x*
VTAB *y*	Move cursor to line *y*
POS (0)	Current horizontal cursor position
INVERSE	Display text in black-on-white
FLASH	Display flashing text
NORMAL	Display text in white-on-black
SPEED= *x*	Set text display rate to *x* (0 minimum, 255 maximum)

Arrays

Type	Typical Element
Real	AB (x, y, z)
Integer	AB% (x, y, z)
String	AB$ (x, y, z)

where A is a letter, B is a letter or digit. Name may be more than two characters, but only first two are significant. Array size limited only by available memory.

DIM a (x, y, z) Define array *a* with maximum subscripts *x, y, z*

Arithmetic Operators

=	Assign value to variable (LET optional)
+	Addition
−	Subtraction
*	
/	Division
^	Exponentiation

Relational Operators

=	Equal to
<	Less than
>	Greater than
<= =>	Less than or equal to
>= =<	Greater than or equal to
<> ><	Not equal to

Yield value 1 if true, 0 if false. Can also be used to compare strings.

Logical Operators

AND	Both true
OR	Either or both true
NOT	Is false

Interpret 0 as false, nonzero as true. Yield value 0 if false, 1 if true.

Precedence of Operators

( )	Parentheses (innermost first)
+ − NOT	Signed arithmetic, logical "not"
^	Exponentiation
* /	Multiplication, division
+ −	Addition, Subtraction
= < > <= =< >= => <> ><	Relational operators
AND	Logical "and"
OR	Logical "or"

Arithmetic Functions

ABS (x)	Absolute value of *x*
SGN (x)	Sign of *x*
INT (x)	Integer part of *x*
SQR (x)	Square root of *x*
SIN (x)	Sine of *x* radians
COS (x)	Cosine of *x* radians
TAN (x)	Tangent of *x* radians
ATN (x)	Arc tangent, in radians, of *x*
EXP (x)	Exponential of *x*
LOG (x)	Natural logarithm of *x*
RND (x)	If $x > 0$, generate random number between 0 and 1 If $x = 0$, repeat previous random number If $x < 0$, begin new repeatable sequence of random numbers
DEF FN (x) = *expr*	*Define function*

Graphics

`GR`	Display low-resolution graphics
`COLOR= x`	Set low-resolution display color to *x*
`PLOT x, y`	Plot single block at column *x*, row *y*
`HLIN x1, x2 AT y`	Draw horizontal line from column x1 to column *x2* in row *y*
`VLIN y1, y2 AT x`	Draw vertical line from row *y1* to row *y2* in column *x*
`SCRN (x, y)`	Color on screen at column *x*, row *y*

Columns numbered from 0 to 39; rows from 0 to 39 in mixed text and graphics, 0 to 47 in full-screen graphics.

`HGR`	Display high-resolution graphics, page 1; mixed text and graphics
`HGR2`	Display high-resolution graphics, page 2; full-screen graphics
`HCOLOR= x`	Set high-resolution display color to *x*
`HPLOT x, y`	Plot single point at column *x*, row *y*
`HPLOT x1, y1 TO x2, y2 TO x3, y3`	Draw high-resolution lines from column *x1*, row *y1* to column *x2*, row *y2* to column *x3*, row *y3*
`HPLOT TO x, y`	Extend previous line to column *x*, row *y*

Columns numbered from 0 to 279; rows from 0 to 159 in mixed text and graphics, 0 to 191 in full-screen graphics.

`SHLOAD`	Load shape table from tape
`DRAW n AT x, y`	Draw shape number *n* at column *x*, row *y*
`XDRAW n AT x, y`	Erase shape number *n* at column *x*, row *y*
`SCALE= x`	Set scale factor for drawing shapes to *x*
`ROT= x`	Set rotation for drawing shapes to *x*

Utility Statements

`PEEK (addr)`	Contents of memory location *addr*
`POKE addr, x`	Store value *x* at memory location *addr*
`CALL addr`	Execute machine-language subroutine starting at location *addr*
`USR (x)`	Execute user-supplied machine-language function routine with argument *x*
`WAIT addr, m1, m2`	Suspend execution until bit pattern specified by masks *m1*, *m2* appears at location *addr*
`HIMEM: addr`	Set highest memory address available for variable storage to *addr*
`LOMEM: addr`	Set lowest memory address available for variable storage to *addr*
`FRE (0)`	Amount of available storage remaining
`TRACE`	Display line number of each statement executed
`NOTRACE`	Stop displaying line number of each statement executed